

# Decentralized Hybrid Formation Control of Unmanned Aerial Vehicles

Ali Karimoddini<sup>1</sup>, Mohammad Karimadini<sup>2</sup>, Hai Lin<sup>3</sup>

**Abstract**—This paper presents a decentralized hybrid supervisory control approach for a team of unmanned helicopters that are involved in a leader-follower formation mission. Using a polar partitioning technique, the motion dynamics of the follower helicopters are abstracted to finite state machines. Then, a discrete supervisor is designed in a modular way for different components of the formation mission including reaching the formation, keeping the formation, and collision avoidance. Furthermore, a formal technique is developed to design the local supervisors decentralizedly, so that the team of helicopters as whole, can cooperatively accomplish a collision-free formation task.

## I. INTRODUCTION

Nowadays, developing Unmanned Aerial Vehicles (UAVs) in different sizes and shapes for various applications has emerged as an attractive research area [1], [2], [3], [4]. A challenging problem in the aerial robotics area and cooperative control of UAVS is *formation control*, in which it is desired to instruct a group of agents to jointly move with a relatively fixed distance. This capability improves the performance of UAVs to accomplish different tasks such as search and coverage more efficiently. In the literature, there are several methods that can partly handle subcomponents of a formation mission. For instance, for *reaching the formation*, methods such as MILP programming, navigation function, and potential field have been developed [5], [6], [7], [8]. *Keeping the formation* can be seen as a standard control problem in which the system's actual position has slightly deviated from the desired position [9], [10], [11]. Finally, in [12], [13], [14], [15], different scenarios for *collision avoidance* have been introduced using geometry approaches, predictive control, probabilistic methods, and invariant sets. Nevertheless, putting all together to address the whole components of the formation mission, requires an in-depth understanding of the interplay between the components based on which a decision making unit can be embedded in the control structure of the UAVs. To make this control structure reliable enough, two main problems should be addressed. Firstly, this control structure has a hybrid nature, which includes both the continuous dynamics of the UAVs and the discrete dynamics of the decision making unit that interactively coexist in the system [16]. Although a common

practice is to treat the continuous and the discrete structure of the system in a decoupled way, the ignorance of the interactions between the continuous and discrete dynamics of the system degrades the reliability of the overall system. Secondly, to take the advantage of decentralized control schemes, e.g. distributing the computation costs among the agents and increasing the reliability of the system against the possible failures, a decentralized controller is required. To address the first problem, in [17], a hybrid supervisory control framework was introduced for the formation control of UAVs.

This paper addresses the second problem and presents a decentralized hybrid supervisory control of UAVs that are involved in a leader-follower formation scenario. First, using the abstraction techniques, a DES model is obtained for the motion dynamics of each agent. Then, the formation task is formulated by logical requirements for which we have modularly designed the discrete supervisors for different components of the formation including reaching the formation, keeping the formation, and collision avoidance. In the reaching and keeping the formation, the follower UAVs can satisfy the desired performance independently. However, for the collision avoidance, a tight cooperation of the UAVs is required. For this purpose, a collision avoidance supervisor is designed, so that the team of UAVs as whole, can cooperatively satisfy the collision avoidance specification as a global goal. Then, to render the decentralized implementation, the designed global supervisor is decomposed into local supervisors through the natural projections into local event sets.

The rest of this paper is organized as follows. Section II describes the problem formulation. Section III obtains an abstract model for the motion dynamics of the follower UAVs using the polar partitioning of the motion space. A discrete supervisor is modularly designed in Section IV, and then, it is decomposed into local supervisors. The paper is concluded in Section VI.

## II. PROBLEM FORMULATION

In [18] and [19] it is shown that subject to the proper implementation of the inner-loop for an unmanned helicopter to be fast enough to track the given references, the outer loop dynamics can be approximately described as follows:

$$\dot{x} = u, \quad x \in \mathbb{R}^2, \quad u \in U \subseteq \mathbb{R}^2, \quad (1)$$

where  $x$  is the position of the UAV;  $u$  is the UAV velocity reference generated by the formation algorithm, and  $U$  is the convex set of velocity constraints.

<sup>1</sup> A. Karimoddini is with the Department of Electrical and Computer Engineering, North Carolina Agricultural and Technical State University, Greensboro, NC 27411 USA, akarimod@ncat.edu.

<sup>2</sup> M. Karimadini is with the Department of Electrical Engineering, Arak University of Technology, Arak, Iran, elekm@nus.edu.sg.

<sup>3</sup> H. Lin is with the Department of Electrical Engineering, University of Notre Dame, Notre Dame, USA, hlin1@nd.edu.

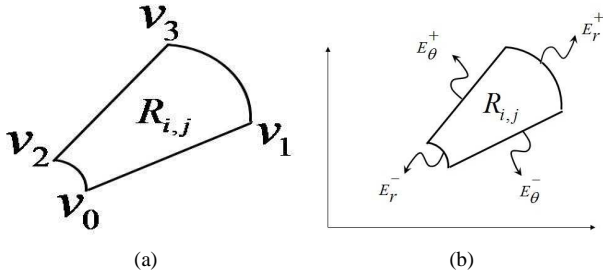


Fig. 1. (a) Vertices of the element  $R_{i,j}$ . (b) Edges of the element  $R_{i,j}$ .

Also, assume that the UAVs are flying at the same altitude, and the velocity of the  $k$ 'th follower,  $UAV_k$ ,  $k = 1, 2$  is in the following form:

$$V_{follower_k} = V_{leader} + V_{rel_k}. \quad (2)$$

Now, we can consider a relatively fixed frame for each follower UAV, in which each follower moves with the relative velocity  $V_{rel}$ .

**Problem 1:** *Given the dynamics of the follower UAVs as (1) and their velocity in the form of (2), design the formation controller to generate the relative velocity of the followers,  $V_{rel_k}$ , such that starting from any initial state inside the control horizon, the follower UAVs eventually reach their desired positions, while avoiding the collision with other follower UAVs. Moreover, after reaching the formation, the follower UAVs should remain at the desired positions.*

### III. DISCRETE MODEL OF THE UAV MOTION DYNAMICS OVER THE POLAR PARTITIONED SPACE

To address this problem, for each UAV consider a circle with the radius of  $R_m$  that is centered at its desired position. With the aid of the partitioning curves  $\{r_i = \frac{R_m}{n_r-1}(i-1), i = 1, \dots, n_r\}$  and  $\{\theta_j = \frac{2\pi}{n_\theta-1}(j-1), j = 1, \dots, n_\theta\}$ , this circle can be partitioned into  $(n_r-1)(n_\theta-1)$  partitioning elements.

In this partitioned space, an element  $R_{i,j} = \{p = (r, \theta) | r_i \leq r \leq r_{i+1}, \theta_j \leq \theta \leq \theta_{j+1}\}$ , has four vertices,  $v_0, v_1, v_2, v_3$  (Fig. 1(a)), four edges,  $E_r^+, E_r^-, E_\theta^+, E_\theta^-$  (Fig. 1(b)). The set  $V(*)$  stands for the vertices that belong to  $*$  ( $*$  can be an edge, or a region  $R_{i,j}$ ).

As shown in [17], for a system with a multi-affine dynamics  $\dot{x} = h(x, u(x))$  defined over this polar partitioned space, two control features can be designed. First, the region  $R_{i,j}$  can be invariant, i.e., the trajectories of the system remain inside the region forever. The other control feature is the exit edge. It is possible to design a controller to drive the system's trajectory to exit from the edge  $E_q^s$ ,  $q \in \{r, \theta\}$  and  $s \in \{+, -\}$ , by choosing the control values  $u(v_m)$  at the vertices. According to the properties of multi-affine systems, the control value at any point inside the region can be achieved based on the control values at the vertices as  $u(x) = \sum_{m=0}^3 \lambda_m(x) u(v_m)$ , which  $\lambda_m(x)$  is a coefficient that determines the weight of  $u(v_m)$  in the control value  $u(x)$ . We denote the controller for having a region invariant by  $C_{0k}$ . Also,  $C_{rk}^+, C_{rk}^-, C_{\theta k}^+$ , and  $C_{\theta k}^-$  are respectively the

controllers for having the edges  $F_r^+, F_r^-, F_\theta^+$ , and  $F_\theta^-$ , as exit edges. Further details on how to design these controllers are provided in [17].

Now, this model of the UAV motion dynamics over the partitioned space can be abstracted to a finite state machine and can be presented by a discrete automaton. An automaton can be formally defined as follows:

**Definition 1:** (Automaton)[20]. A deterministic automaton is a tuple  $A := (Q, q_0, E, \delta, Q_m)$  consisting of a set of states  $Q$ ; an initial state  $q_0 \in Q$ ; a set of events  $E$  that causes transitions between the states, and a transition relation  $\delta \subseteq Q \times E \times Q$  (with a partial map  $\delta : Q \times E \rightarrow Q$ ), such that  $(q, e, q') \in \delta$  if and only if state  $q$  is transited to state  $q'$  by event  $e$ , denoted by  $q \xrightarrow{e} q'$  (or  $\delta(q, e) = q'$ ).  $Q_m \subseteq Q$  represents the marked states to assign a meaning of accomplishment to some states. For supervisor automaton whose all states are marked,  $Q_m$  is omitted from the tuple.

For this automaton, the sequence of these events forms a string. We use  $\varepsilon$  to denote an empty string, and  $\Sigma^*$  to denote the set of all possible strings over the set  $\Sigma$  including  $\varepsilon$ . The language of the automaton  $G$ , denoted by  $L(G)$ , is the set of all strings that can be generated by  $G$ , starting from the initial states. The marked language,  $L_m(G)$ , is the set of strings that belong to  $L(G)$  and end with the marked states.

For  $UAV_1$ , the discrete model of the system over the partitioned space can be described by the automaton  $A_1 = (Q_1, q_{01}, E_1, \delta_1, Q_{m1})$  whose set of discrete states is  $Q_1 = \{R_1, O_1\}$ , and its event set is  $E_1 = C_1 \cup \{C_{01}\} \cup D_1 \cup Ex$ , where  $C_1 = \{C_{r1}^+, C_{r1}^-, C_{\theta1}^+, C_{\theta1}^-\}$  and  $D_1 = \{d_{i,j1} | 1 \leq i \leq n_r - 1, 1 \leq j \leq n_\theta - 1\}$ . When  $UAV_1$  is in one of the regions  $R_{i,j}$ , in the abstract model it is considered to be in the discrete state  $R_1$ . Then, one of the actuation commands belong to  $C_1$  drives the UAV to one of its adjacent regions. In this case, right after issuing the actuation commands, the system transits to the detection state  $O_1$  and waits until the UAV enters a new region. Crossing boundaries of the new region, a detection event belonging to  $D_1$  will be generated which shows the UAV has entered the new region  $R_{i',j'}$ . The command  $C_{01}$ , keeps the UAV in the current region and does not change the discrete state of the system. We use the notation  $D_{M1} = \{d_{i,j1} | 1 \leq i = 1, 1 \leq j \leq n_\theta - 1\}$  to denote the detection events, which show entering a region in the first circle, and  $d_1 = D_1 - D_{M1} = \{d_{i,j1} | 1 < i \leq n_r - 1, 1 \leq j \leq n_\theta - 1\} \subseteq D_1$  for the rest of detection events. Here,  $Ex = \{Ca_{12F}, Ca_{12N}, Ca_{21F}, Ca_{21N}, Stop_1, Stop_2, R_{21}, R_{12}\}$  is the set of external events which are required for the collision avoidance and do not change the state of the system. The events belong to  $CA = \{Ca_{12F}, Ca_{12N}, Ca_{21F}, Ca_{21N}\}$  show the collision alarms, in which the events in  $CA_1 = \{Ca_{12F}, Ca_{12N}\}$  show that  $UAV_2$  enters the alarm zone of  $UAV_1$  and accordingly, the events in  $CA_2 = \{Ca_{21F}, Ca_{21N}\}$  show that  $UAV_1$  enters the alarm zone of  $UAV_2$ . The details will be discussed in Section IV-B. The events  $Stop_1$  and

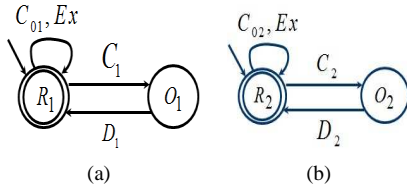


Fig. 2. (a) DES model of  $UAV_1$ . (b) DES model of  $UAV_2$ .

$Stop_2$  are the commands that request  $UAV_1$  and  $UAV_2$  to stop at their current position in the relative frame and the command  $R_{12}$  and  $R_{21}$  release them, respectively. Similar definitions can be given for the DES model of  $UAV_2$ . The graph representation of the discrete models of  $UAV_1$  and  $UAV_2$  are shown in Fig. 2. In these graphs, the arrows starting from one state and ending to another state represent the transitions, labeled by the events belong to  $E_i$ . The entering arrows stand for the initial states. Marked states are shown by double circles.

In the DES model of  $UAV_k$ ,  $k = 1, 2$ , the event set  $E_k$  consists of the controllable event set  $E_{ck} = \{C_{0k}, C_{r_k}^+, C_{r_k}^-, C_{\theta_k}^+, C_{\theta_k}^-, Stop_1, Stop_2, R_{21}, R_{12}\}$  and the uncontrollable event set  $E_{uck} = \{Ca_{12F}, Ca_{12N}, Ca_{21F}, Ca_{21N}\} \cup D_1$ . The uncontrollable events are those that cannot be affected by the supervisor. A language  $K$  is controllable with respect to the language  $L(A)$  and the event set  $E_{uc}$  if and only if  $\forall s \in K$  and  $\sigma \in E_{uc}$ , if  $s\sigma \in L(A)$ , then  $s\sigma \in K$ . Indeed, the controllability is the existence condition of a supervisor for the control goal described by the specification  $K$  [20].

#### IV. DESIGNING A DECENTRALIZED MODULAR SUPERVISOR FOR THE FORMATION CONTROL OF THE UAVS

Given the discrete model of follower UAVs over the partitioned space, it is possible to design the supervisor to achieve a desired order of events to accomplish the formation. Indeed, the supervisor,  $S$ , observes the executed strings of the plant  $A$  and disables the undesirable controllable events. Here, we assume that all of the events are observable. The generated language and marked language of the closed-loop system,  $L(S/A)$  and  $L_m(S/A)$ , can be constructed as follows:

- (1)  $\varepsilon \in L(S/A)$
- (2)  $[(s \in L(S/A)) \text{ and } (s\sigma \in L(A)) \text{ and } (\sigma \in L(S))] \Leftrightarrow (s\sigma \in L(S/A))$
- (3)  $L_m(S/A) = L(S/A) \cap L_m(A)$

where  $s$  is the string that has been generated so far by the plant  $A$ , and  $\sigma$  is an event, which the supervisor  $S$  should decide whether keep it active or not in the supervised system  $S/A$ .

Within this framework one can use parallel composition to facilitate the control synthesis. Parallel composition is a binary operation between two automata which can be defined as follows:

**Definition 2:** (Parallel Composition [20]) Let  $A_i = (Q_i, q_i^0, E_i, \delta_i, Q_{m_i})$ ,  $i = 1, 2$ , be automata. The paral-

lel composition (synchronous composition) of  $A_1$  and  $A_2$  is the automaton  $A_1 \parallel A_2 = (Q = Q_1 \times Q_2, q_0 = (q_1^0, q_2^0), E = E_1 \cup E_2, \delta, Q_m = Q_{m_1} \times Q_{m_2})$ , with  $\delta$  defined as  $\forall (q_1, q_2) \in Q, e \in E: \delta((q_1, q_2), e) =$

$$\begin{cases} (\delta_1(q_1, e), \delta_2(q_2, e)), & \text{if } \delta_1(q_1, e)!, \delta_2(q_2, e)!, \\ & e \in E_1 \cap E_2; \\ (\delta_1(q_1, e), q_2), & \text{if } \delta_1(q_1, e)!, e \in E_1 \setminus E_2; \\ (q_1, \delta_2(q_2, e)), & \text{if } \delta_2(q_2, e)!, e \in E_2 \setminus E_1; \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

Here, the parallel composition is used to combine the plant's discrete model and the supervisor as follows:

**Lemma 1:** [21] Let  $A = (Q, q_0, E, \alpha, Q_m)$ , be the plant automaton and  $K \subseteq E^*$  be the desired marked language. There exists a nonblocking supervisor  $S$  such that  $L_m(S/A) = L_m(S \parallel A) = K$  if  $\emptyset \neq K = \bar{K} \cap L_m(A)$  and  $K$  is controllable. In this case,  $S$  could be any automaton with  $L(S) = L_m(S) = \bar{K}$ .

Now, using the above lemma, it is possible to design the supervisor for the formation problem described in Problem 1, which includes two modules: 1- Reaching and keeping the formation and 2- Avoiding collision. Next lemma describes how to design the supervisors in a modular way.

**Lemma 2:** [21] Let  $A = (Q, q_0, E, \alpha, Q_m)$  be the plant automaton and the prefix-closed controllable languages  $K_1, K_2 \subseteq E^*$  be the desired marked specifications. Suppose there exist nonblocking supervisors  $S_1$  and  $S_2$  such that  $L_m(S_1/A) = L_m(S_1 \parallel A) = K_1$  and  $L_m(S_2/A) = L_m(S_2 \parallel A) = K_2$ , then  $S = S_1 \parallel S_2$  is a nonblocking supervisor with  $L_m(S \parallel A) = K_1 \cap K_2$ . ■

#### A. Designing the supervisor for reaching and keeping the formation

For reaching the formation, it is sufficient to directly drive each of the follower UAVs towards one of the regions  $R_{1,j}$ ,  $1 \leq j \leq n_\theta - 1$ , located in the first circle in their corresponding partitioned motion space. After reaching  $R_{1,j}$ , the UAVs should remain inside it, to keep the formation. The specifications  $K_{F1}$  and  $K_{F2}$  for reaching and keeping the specification for  $UAV_1$  and  $UAV_2$  are realized in Fig. 3. When the  $k$ 'th follower UAV is not in the first circle, the command  $C_{r_k}^-$  will be generated to push the UAV towards the origin. Entering a new region, one of the events from  $d_k = \{d_{i,j_k} \mid 1 < i \leq n_r - 1, 1 \leq j \leq n_\theta - 1\}$  will appear. This will continue until one of the events from  $D_{Mk} = \{d_{i,j_k} \mid i = 1, 1 \leq j \leq n_\theta - 1\}$  be generated, which shows that the formation is reached. In this case, the event  $C_{0k}$  is activated, which keeps the system trajectory inside the first region. If a collision alarm happens to  $UAV_k$ , the formation supervisor does not change the generable language after the events belonging to  $CA$ , and lets the collision avoidance supervisor handle it until the collision be avoided and the UAV be released to resume the formation task.

It can be seen that  $K_{Fk}$ ,  $k = 1, 2$  are controllable with respect to the plant language  $L(A_k)$  and the event set  $E_{uck}$ , as they do not disable any uncontrollable event. Therefore, based on Lemma 1, there exist supervisors that can control the plants  $A_1$  and  $A_2$  to achieve these specifications. The

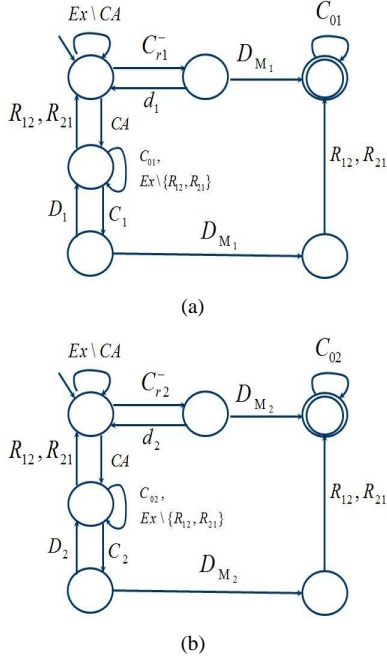


Fig. 3. (a) The specification for reaching and keeping the formation for UAV1. (b) The specification for reaching and keeping the formation for UAV2.

supervisors are the realization of the above specifications in which all states are marked. Marking all states of the supervisors allows the closed-loop marked states to be solely determined by the plants' marked states. The supervisor for reaching the formation and keeping the formation of  $UAV_k$  is denoted by  $A_{F_k}$ .

### B. Designing the supervisor for collision avoidance

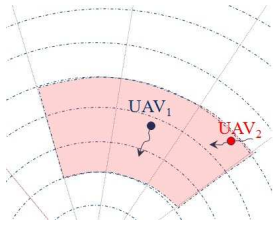


Fig. 4.  $UAV_2$  enters the alarm zone of  $UAV_1$ .

When  $UAV_1$  is going to reach its desired position, in some situations, the other follower,  $UAV_2$ , may enter the alarm zone of  $UAV_1$  (Fig. 4), which requires these UAVs to cooperatively avoid the collision. For this purpose, first,  $UAV_1$  asks  $UAV_2$  to stop in the relative frame and then,  $UAV_1$  finds a path to safely get away from  $UAV_2$ . After avoiding the collision,  $UAV_1$  releases  $UAV_2$  and both UAVs resume their normal operation for reaching the formation. Similar strategy is taken when  $UAV_1$  enters the alarm zone of  $UAV_2$ . This specification,  $K_C$ , is shown in Fig. 5 whose

left side shows that after appearing one of the events  $ca_{12F}$  or  $ca_{12N}$ ,  $UAV_1$  realizes that  $UAV_2$  has entered its alarm zone. Therefore, by event  $Stop_2$ ,  $UAV_1$  requests  $UAV_2$  to stop for a while to safely manage the situation. The event  $ca_{12F}$  shows that  $UAV_2$  is in front of the path of  $UAV_1$  towards its destination and hence, to avoid the collision it is sufficient that  $UAV_1$  turns anticlockwise to change its azimuth angle,  $\theta$ , by activating the command  $C_\theta^+$ . This will continue until removing the collision alarm. Then,  $UAV_1$  releases  $UAV_2$ , and reaching the formation can be resumed by the reaching formation supervisor which was explained in the previous section. Meanwhile, if  $UAV_1$  enters one of the regions in the first circle, one of the events belong to  $D_{M1}$  appears which means that  $UAV_1$  has reached its desired formation and should remain there for the rest of mission. Similarly, the right side of Fig. 5, shows the collision avoidance mechanism when  $UAV_1$  enters the alarm zone of  $UAV_2$ . If neither of collision avoidance alarms from the set  $CA$  happens, then  $UAV_1$  and  $UAV_2$  can do their normal operations by independent enabling of events  $C_1$  and  $C_2$  followed by the detection signals  $D_1$  and  $D_2$  in any order as shown on the top of Fig. 5. The other module, the reaching formation supervisor, will manage this situation.

It can be verified that  $K_C$  is controllable with respect to the language  $L(A_1 || A_2)$  and the event set  $E_{uc1} \cup E_{uc2}$ . Therefore, based on Lemma 1, there exists a supervisor  $A_c$  that can control the plants  $A_1$  and  $A_2$  to achieve this joint specification. The supervisor is the realization of the specification  $K_C$  in which all states are marked.

The collision avoidance supervisor,  $A_C$ , is a centralized supervisor which manages both  $UAV_1$  and  $UAV_2$ . To make this supervisor decentralized and to achieve local supervisors, we will utilize our proposed decomposition scheme introduced in [22]. Here, local supervisors can be achieved by the projection of the global supervisor to each agent's local event set. The projection of the global supervisor  $A_C$  to the event set of  $UAV_i$ ,  $E_i$ , is denoted by  $P_{E_i}(A_C)$ , and can be obtained by replacing the events that belong to  $E \setminus E_i$  by  $\varepsilon$ -moves, and then, merging the  $\varepsilon$ -related states.

Once the local supervisor automata are derived through the natural projection, the decentralized supervisor is then obtained using the parallel composition of local supervisor automata. Parallel composition captures the logical behavior of concurrent distributed systems by allowing each subsystem to evolve individually on its private events, while synchronize with its neighbors on shared events for cooperative tasks.

The obtained decentralized supervisor is then compared with the original global supervisor automaton using the bisimulation relation.

**Definition 3:** Consider two automata  $A_i = (Q_i, q_i^0, E, \delta_i)$ ,  $i = 1, 2$ . The automaton  $A_1$  is said to be similar to  $A_2$  (or  $A_2$  simulates  $A_1$ ), denoted by  $A_1 \prec A_2$ , if there exists a relation  $R$  from  $A_1$  to  $A_2$  over  $Q_1, Q_2$  and with respect to  $E$ , such that (1)  $(q_1^0, q_2^0) \in R$ , and (2)  $\forall (q_1, q_2) \in R, q_1' \in \delta_1(q_1, e)$ , then  $\exists q_2' \in Q_2$  such that  $q_2' \in \delta_2(q_2, e)$ ,  $(q_1', q_2') \in R$ . Automata  $A_1$  and  $A_2$



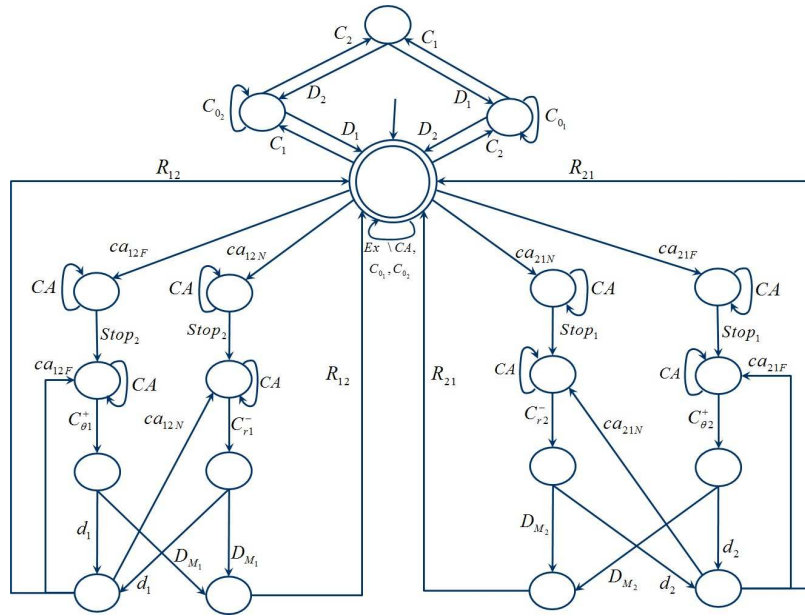


Fig. 5. The specification for cooperative collision avoidance.

are said to be bisimilar (bisimulate each other), denoted by  $A_1 \cong A_2$  if  $A_1 \prec A_2$  with a simulation relation  $R_1$ ,  $A_2 \prec A_1$  with a simulation relation  $R_2$  and  $R_1^{-1} = R_2$ , where  $R_1^{-1} = \{(y, x) \in Q_2 \times Q_1 | (x, y) \in R_1\}$ .

Based on these definitions we can formally describe the decomposability conditions with respect to two local event sets.

**Lemma 3:** (Theorem 4 in [22]) A deterministic automaton  $A = (Q, q_0, E = E_1 \cup E_2, \delta)$  is decomposable with respect to parallel composition and natural projections  $P_i$ ,  $i = 1, 2$ , such that  $A \cong P_1(A) \parallel P_2(A)$  if and only if  $A$  satisfies the following decomposability conditions (DC):  $\forall e_1 \in E_1 \setminus E_2, e_2 \in E_2 \setminus E_1, q \in Q, s \in E^*$ ,

- DC1:  $[\delta(q, e_1)! \wedge \delta(q, e_2)!] \Rightarrow [\delta(q, e_1 e_2)! \wedge \delta(q, e_2 e_1)!]$ ;
- DC2:  $\delta(q, e_1 e_2 s)! \Leftrightarrow \delta(q, e_2 e_1 s)!;$
- DC3:  $\forall s, s' \in E^*, s \neq s', p_{E_1 \cap E_2}(s), p_{E_1 \cap E_2}(s')$  start with the same common event  $a \in E_1 \cap E_2$ ,  $q \in Q$ :  $\delta(q, s)! \wedge \delta(q, s')! \Rightarrow \delta(q, p_1(s)|p_2(s'))! \wedge \delta(q, p_1(s')|p_2(s))!;$
- DC4:  $\forall i \in \{1, 2\}, x, x_1, x_2 \in Q_i, x_1 \neq x_2, e \in E_i, t \in E_i^*, x_1 \in \delta_i(x, e), x_2 \in \delta_i(x, e): \delta_i(x_1, t)! \Leftrightarrow \delta_i(x_2, t)!.$

where,  $\bar{K} = \{s \in \Sigma^* | (\exists t \in \Sigma^*) st \in K\}$  is the prefix closure of the language  $K$ . The decomposability conditions DC1 and DC2 respectively guarantee that any decision on the selection or order of two transitions can be done by the team of agent, while conditions DC3 and DC4 respectively ensure that the interaction of local automata  $P_1(A)$  and  $P_2(A)$  neither allows an illegal string that is not in  $A$ , nor stops a legal string of  $A$ .

Now assume that given the global task and local plants, a global supervisor is designed and decomposed into local supervisors such that each closed loop system (the supervised local plant with the corresponding local controller) satisfies the global task. In this decentralized cooperative control

architecture we are then interested to check whether the entire system satisfied the global task.

**Problem 2:** (Decentralized cooperative control problem) Consider a plant, represented by a parallel distributed system  $A_P := \parallel_{i=1}^2 A_{P_i}$ , with local event sets  $E_i$ ,  $i = 1, 2$ , and let the global specification is given by a deterministic task automaton  $A_S$  over  $E = \bigcup_{i=1}^2 E_i$ . Furthermore, suppose that there exist a decomposable deterministic global controller automaton  $A_C \cong \parallel_{i=1}^2 P_i(A_C)$ , so that  $A_P \parallel A_C \cong A_S$ . Then, whether the local controllers can lead the team to satisfy the global specification in a decentralized architecture,  $\parallel_{i=1}^2 (A_{P_i} \parallel P_i(A_C)) \cong A_S$ .

Following result considers a team of two local plants and introduces the supervisor decomposability and satisfaction of the global task by each local supervised plant as a sufficient condition for the satisfaction of global task by the team.

**Theorem 1:** (Decentralized cooperative control using supervisor decomposition) Consider a plant, represented by a parallel distributed system  $A_{P_1} \parallel A_{P_2}$ , with local event sets  $E_i$ ,  $i = 1, 2$ , and let the global specification is given by a task automaton  $A_S$  over  $E = E_1 \cup E_2$ . Furthermore, suppose that there exist a deterministic global controller automaton  $A_C \cong P_1(A_C) \parallel P_2(A_C)$ , so that  $A_C \parallel A_P \cong A_S$ . Then, the entire closed loop system satisfies the global specification, in the sense of bisimilarity, i.e.,  $\parallel_{i=1}^2 (A_{P_i} \parallel P_i(A_C)) \cong A_S$ , provided the decomposability conditions DC1, DC2, DC3 and DC4 for  $A_C$ .

The significance of this result is the decentralized implementation of the global supervisor,  $A_C$ , given in Fig. 5, by decomposing  $A_C$ , into local supervisors. As it can be

seen in  $A_C$ , the successive and adjacent events from pairs of private event sets (from different local event sets)  $(C_{01}, C_2)$ ,  $(C_{01}, D_2)$ ,  $(C_{02}, C_1)$ ,  $(C_{02}, D_1)$ ,  $(C_1, D_2)$ ,  $(C_2, D_2)$ , appear in both orders in the global supervisor automaton therefore  $DC1$  and  $DC2$  are satisfied. Moreover, among common events  $R_{12}$ ,  $R_{21}$ ,  $CA_1 = \{ca_{12F}, ca_{12N}\}$ ,  $CA_2 = \{ca_{21F}, ca_{21N}\}$ ,  $Stop_1$ , and  $Stop_2$ , the events  $R_{12}$ ,  $R_{21}$ ,  $Stop_1$ , and  $Stop_2$  are not shared between different strings. Strings just share the events  $CA_1$ ,  $CA_2$ , where the corresponding local strings do not interleave on these events because of predecessor common events before  $CA_1$ ,  $CA_2$ . Therefore  $DC3$  also is fulfilled. Finally,  $DC4$  is satisfied because of the determinism of local automata  $P_1(A_C)$  and  $P_2(A_C)$ , and hence, the supervisor automaton  $A_C$  is decomposable into  $A_{C1} = P_1(A_C)$  and  $A_{C2} = P_2(A_C)$ , shown in Fig. 6, so that  $A_{C1} \parallel A_{C2} \cong A_C$ .

## V. VERIFYING THE ALGORITHM THROUGH A HARDWARE-IN-THE-LOOP SIMULATION PLATFORM

To verify the proposed algorithm, we have used a hardware-in-the-loop simulation platform [23] developed for NUS UAV helicopters [24]. In this platform, the nonlinear dynamics of the UAVs have been replaced with their nonlinear model, and all software and hardware components that are involved in a real flight test remain active during the simulation so that the simulation results achieved from this simulator are very close to the actual flight tests. This multi-UAV simulator test bed is used to verify the proposed algorithm. For this purpose, consider two followers that should track a leader UAV with a desired distance, as shown in Fig. 7. The distance between the desired position of the *Follower<sub>1</sub>* and *Follower<sub>2</sub>* and the leader UAV are  $\Delta_{d1} = (12, 10)$  and  $\Delta_{d2} = (-12, -10)$ , respectively. The follower UAVs initially are not at the desired position. The initial distance between *Follower<sub>1</sub>* and its desired position is  $\Delta_{01} = (-41.9, -0.9)$ , and the initial distance between *Follower<sub>2</sub>* and its desired position is  $\Delta_{02} = (-17.5, 0.5)$ . *Follower<sub>1</sub>* after 34.8 sec and *Follower<sub>2</sub>* after 14.3 sec reach the formation and then, they will keep the formation.

After 50 sec, the formation switches. For the new formation, the desired distance of the followers from the leader are  $\Delta_{d1} = (-30, -10)$  and  $\Delta_{d2} = (0, 10)$ , while their initial distances from the desired position are  $\Delta_{01} = (40.5, 23.3)$  and  $\Delta_{02} = (-14.5, -23)$ . When the followers are trying to reach the desired formation, at  $t = 55.8$  sec, *Follower<sub>2</sub>* enters the alarm zone of *Follower<sub>1</sub>*. As described in Section IV-B, to avoid collision, *Follower<sub>1</sub>* asks *Follower<sub>2</sub>* to stop in the relative frame, and then it turns to handle the situation. After removing the collision alarm, both followers have resumed their normal operation to reach and keep the formation. The indices of the traversed regions for  $\theta$  and  $r$  are shown in Fig. 9. The position of the UAVs in x-y plane is shown in Fig. 8.

## VI. CONCLUSION

In this paper, a collision free formation control algorithm was proposed using hybrid supervisory control techniques.

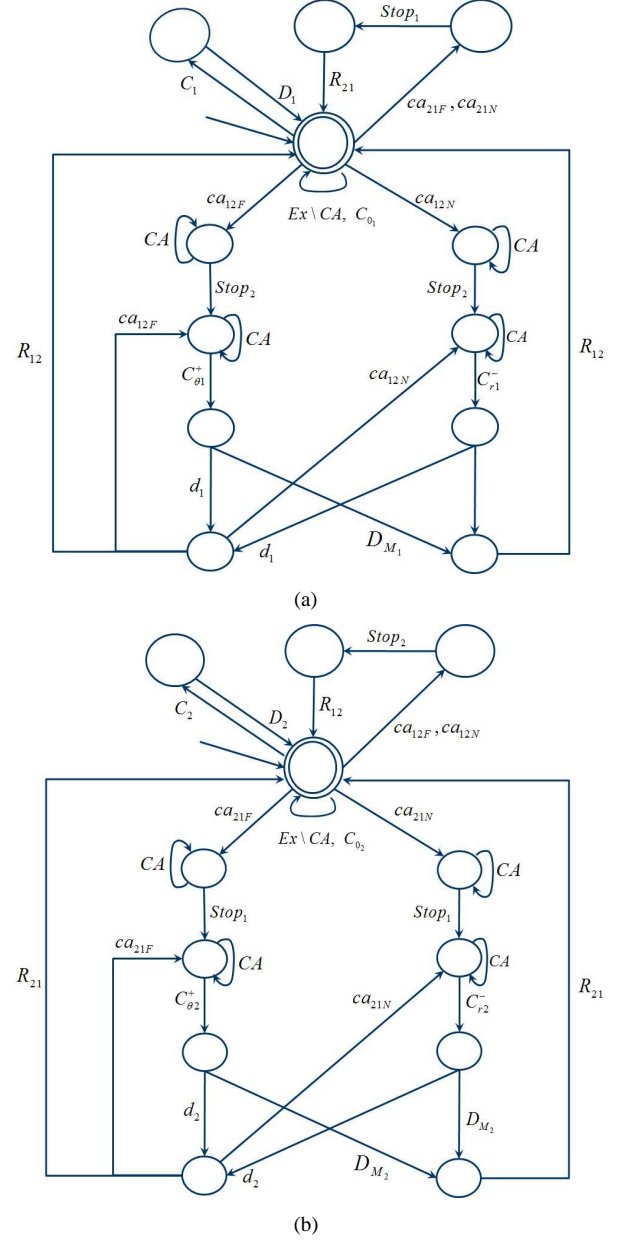


Fig. 6. (a) The local supervisor for collision avoidance for *UAV<sub>1</sub>*. (b) The local supervisor for collision avoidance for *UAV<sub>2</sub>*.

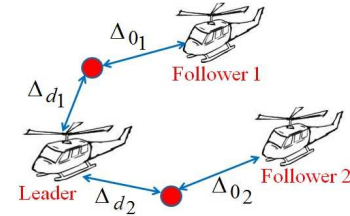


Fig. 7. The schematic of a formation scenario with two followers and one leader

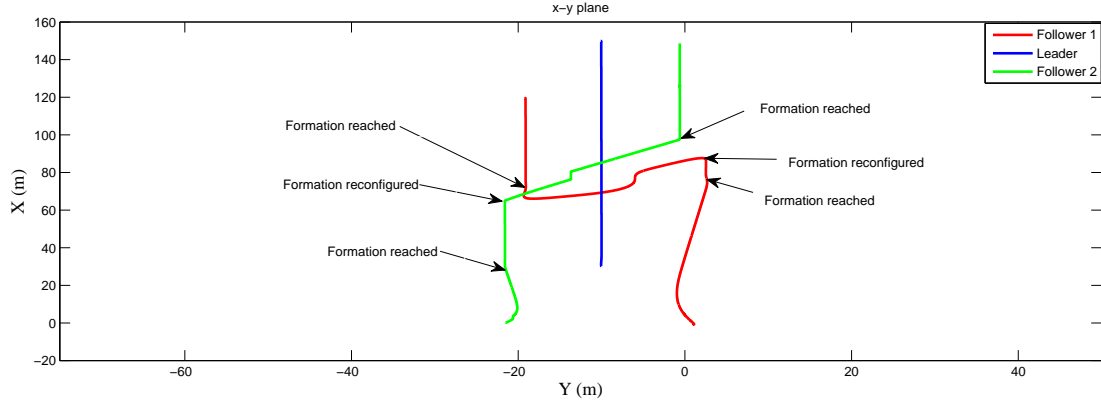


Fig. 8. The position of the UAVs in the x-y plane.

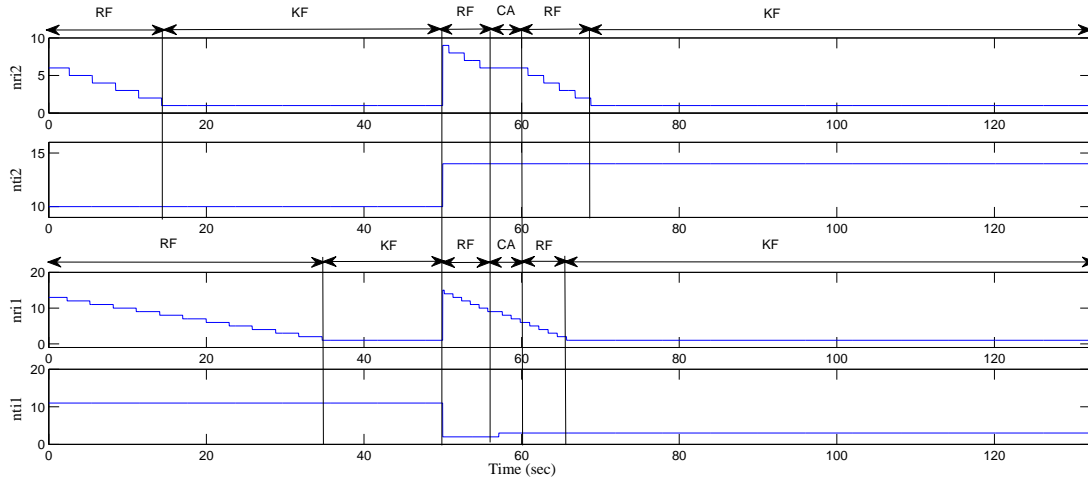


Fig. 9. The indices of  $\theta$  and  $r$  for the traversed regions by *Follower*<sub>1</sub> and *Follower*<sub>2</sub>.

The proposed supervisor has a modular structure and can accomplish three main tasks: reaching the formation, keeping the formation, and collision avoidance. This control structure was implemented decentralizedly so that local (decomposed) supervisors can treat the distributed agents to achieve a globally safe and collision free environment. The efficiency of the proposed approach was verified through hardware-in-loop simulation results.

#### ACKNOWLEDGMENT

The financial supports from NSF-CNS-1239222 and NSF-EECS-1253488 for this work are greatly acknowledged.

#### REFERENCES

- [1] K. P. Valavanis, K. P. Valavanis, *Advances in unmanned aerial vehicles: state of the art and the road to autonomy*, Springer Publishing Company, Incorporated, 2007.
- [2] S. A. Bortoff, The university of toronto rc helicopter: a test bed for nonlinear control, in: *Control Applications*, 1999. Proceedings of the 1999 IEEE International Conference on, Vol. 1, IEEE, 1999, pp. 333–338.
- [3] R. C. Michelson, S. Reece, Update on flapping wing micro air vehicle research-ongoing work to develop a flapping wing, crawling entomopter, in: *13th Bristol International RPV/UAV Systems Conference Proceedings*, Bristol England, Vol. 30, 1998, pp. 30–1.
- [4] A. R. Partovi, H. Lin, G. Cai, B. Chen, A. Kevin, Development of a cross style quadrotor, in: *AIAA Guidance, Navigation, and Control Conference*, 2012.
- [5] N. Leonard, E. Fiorelli, Virtual leaders, artificial potentials and coordinated control of groups, in: *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, 2001.
- [6] D. E. Koditschek, E. Rimon, Robot navigation functions on manifolds with boundary, *Adv. Appl. Math.* 11 (1990) 412–442.
- [7] M. Mukai, T. Azuma, M. Fujita, A collision avoidance control for multi-vehicle using pwa/mld hybrid system representation, in: *Control Applications*, 2004. Proceedings of the 2004 IEEE International Conference on, Vol. 2, 2004, pp. 872 – 877. doi:10.1109/CCA.2004.1387478.
- [8] A. Richards, J. How, Aircraft trajectory planning with collision avoidance using mixed integer linear programming, in: *American Control Conference*, Vol. 3, 2002, pp. 1936–1941.
- [9] D. M. Stipanovic, G. Inalhan, R. Teo, C. J. Tomlin, Decentralized overlapping control of a formation of unmanned aerial vehicles, *Automatica* 40 (8) (2004) 1285 – 1296.
- [10] G. Hassan, K. Yahya, I. ul Haq, Leader-follower approach using full-state linearization via dynamic feedback, in: *International Conference on Emerging Technologies*, 2006, pp. 297–305.
- [11] F. Giulietti, M. Innocenti, M. Napolitano, L. Pollini, Dynamic and control issues of formation flight, *Aerospace Science and Technology* 9 (1) (2005) 65–71.
- [12] J. W. Park, H. D. Oh, M. J. Tahk, Uav collision avoidance based on geometric approach, in: *SICE Annual Conference*, 2008, pp. 2122–2126.
- [13] E. Boivin, A. Desbiens, E. Gagnon, Uav collision avoidance using

- cooperative predictive control, in: Control and Automation, 2008 16th Mediterranean Conference on, 2008, pp. 682–688.
- [14] K. Y. Kim, J. W. Park, M. J. Tahk, Uav collision avoidance using probabilistic method in 3-d, in: Control, Automation and Systems, 2007. ICCAS '07. International Conference on, 2007, pp. 826 –829. doi:10.1109/ICCAS.2007.4407015.
  - [15] F. Borrelli, T. Keviczky, G. Balas, Collision-free uav formation flight using decentralized optimization and invariant sets, in: Decision and Control, 2004. CDC. 43rd IEEE Conference on, Vol. 1, 2004, pp. 1099–1104.
  - [16] P. J. Antsaklis, J. A. Stiver, M. Lemmon, Hybrid system modeling and autonomous control systems, in: Hybrid Systems, Springer, 1993, pp. 366–392.
  - [17] A. Karimoddini, H. Lin, B. M. Chen, T. H. Lee, Hybrid formation control of the unmanned aerial vehicles, *Mechatronics* 21 (5) (2011) 886–898.
  - [18] A. Karimoddini, G. Cai, B. M. Chen, H. Lin, T. H. Lee, Hierarchical Control Design of a UAV Helicopter,” in *Advances in Flight Control Systems*, INTECH, Vienna, Austria, 2011.
  - [19] A. Karimoddini, G. Cai, B. M. Chen, H. Lin, T. H. Lee, Multi-layer flight control synthesis and analysis of a small-scale uav helicopter, in: *IEEE Conference on Robotics Automation and Mechatronics*, 2010, pp. 321–326.
  - [20] C. G. Cassandras, S. Lafortune, *Introduction to discrete event systems*, Springer, 2008.
  - [21] R. Kumar, V. K. Garg, *Modeling and Control of Logical Discrete Event Systems*, Vol. 300 of The Springer International Series in Engineering and Computer Science, Springer, 1995.
  - [22] M. Karimadini, H. Lin, Guaranteed global performance through local coordinations, *Automatica* 47 (5) (2011) 890–898.
  - [23] G. Cai, B. M. Chen, T. H. Lee, M. Dong, Design and implementation of a hardware-in-the-loop simulation system for small-scale uav helicopters, *Mechatronics* 19 (7) (2009) 1057–1066.
  - [24] K. Peng, G. Cai, B. M. Chen, M. Dong, K. Y. Lum, T. H. Lee, Design and implementation of an autonomous flight control law for a uav helicopter, *Automatica* 45 (10) (2009) 2333 – 2338.